IMPLEMENTASI PENGEMBANGAN PENJADWALAN *ROUND-ROBBIN* PADA ANTRIAN DATA *REAL TIME CENTRAL PROCESSING UNIT*

Folkes E. Laumal
Politeknik Negeri Kupang, Jl. Adicucipto – Penfui - Kupang, Kode Pos 85361
Email : folkeslaumal76@gmail.com

ABSTRAK

Penjadwalan merupakan dasar sistem operasi komputer multiprogrammingyang membuat sistem menjadi lebih produktif dan efisien. Tujuan dari penjadwalan proses adalah meningkatkan kinerja sistem komputer menurut kriteria tertentu diantaranya waiting timedanturn around time.Real time fetching adalah proses pengambilan data dari antrian pada memory secara real time[1]. Salah satu penjadwalan yang dapat diterapkan dalam pengolahan sistem operasi yaitu Round Robbin. Penjadwalan ini bersifat preemtive sehingga ketika diterapkan pada real time fetching diperoleh hasil performance yang rendah[5]. Untuk itu telah dikembangkan lagi sebuah algoritma Round Robbin (Pengembangan Round Robbin) dan diuji dalam 5 antrian program. Hasil akhir membuktikan bahwa dengan algoritma pengembangan Round Robbin tersebut telah memperkecil waiting time (AWT) dari 7.8 ke 3.6 satuan waktu dan 8.6 ke 6 satuan waktu. Turn around time(ATAT) berkurang dari 13.4 ke 8 satuan waktu dan 12.2 ke 11.6 satuan waktu. Sedangkan performance CPU telah meningkat dari 0.128 ke 0.278 dan dari 0.116 ke 0.167.

Kata kunci : Round-Robbin, waiting time, turn arround time, Performance CPU

1. PENDAHULUAN

Penjadwalan merupakan dasar dari sistem operasi (OS) dalam *multiprogramming computer* karena dengan mengatur aliran dari proses-proses yang ada pada sebuah antrian sistem operasi akan membuat sistem komputer menjadi lebih produktif dan efisien. Sasaran *multiprogramming* adalah mempunyai proses yang tereksekusi di setiap waktu sehingga *utilisasi*dari pemroses menjadi lebih baik. Untuk sistem komputer dengan pemroses tunggal (*single processor*), tidak pernah lebih dari satu proses yang berjalan.

Di dalam sebuah sistem komputer, jika terdapat beberapa proses,maka satu proses akan berjalan dan proses lain akan menunggu sampai proses sebelumnya selesai dieksekusi. Ide *multiprogramming*sebenarnya sederhana karena satu proses dieksekusi sampai selesai tanpa menunggu, biasanya terjadi dalam operasi I/O.

Pada *multiprogramming*, dalam suatu waktu tertentu beberapa proses akan disimpan dimemori. Ketika proses tertentu harus menunggu dalam antrian, setiap kali satu proses harus menunggu, proses lain akan mengambil alih penggunaan pemroses. Dalam hal ini, sistem operasi mengambil pemroses darinya dan memberikan pemroses ke proses lain. Pola ini dilakukan secara terus-menerus.

Tujuan utama dari penjadwalan proses adalah meningkatkan kinerja sistem komputer menurut kriteria tertentu. Kinerja untuk mengukur dan optimasi kerja penjadwalan adalah *fairness*, *response time*, *turn around time* dan *throughput*[3].

Terdapat 2 model penjadwalan dalam sistem operasi yaitu penjadwalam *preemtive*dan *non-preemtive*. Masing-masing penjadwalan itu memiliki sub model dengan spesifikasi dan implementasinya masing-masing, termasuk *Round Robin Scedulling* yang merupakan salah satu penjadwalan *preemtive* yang tanpa prioritas [5]. Penjadwalan ini banyak dikembangkan dalam sistem operasi modern saat ini ketika sebuah antrian dalam pemrosesan data CPU terjadi. Akan tetapi sebenarnya penjadwalan Round robin memiliki keterbatasan jika diterapkan pada sistem antrian yang *real time*. Pada kondisi ini antrian yang datang secara terus-menerus menyebabkan *time sharring* akan bertambah dan menyebabkan menurunnya *performance* dari CPU.

Untuk memperbaiki penurunan performance ini, Ajit Singh, Priyanka Goyal dan Sahil Bahtra (2012) telah berhasil menurunkan jumlah switching dan rata-rata waktu tunggu dengan pendekatan algoritma lain yang diuji pada kasus 5 buah antrian. Akan tetapi seberapa besar penurunan pada rata-rata waktu tunggu dan rata-*rata turn arround time* sehingga dapat memberikan kesimpulan yang pasti tentang perbaikan performance ini, maka

penulis telah menguji kembali melalui sebuah algoritma pengembangan Round Robbin dan menerapkan pada 5 antrian aritmetika pada CPU. Selanjutnya menghitung AWT dan ATAT dengan memperhitungkan *arrival time* dan tanpa *arrival time*. Hasil AWT dan ATAT ini kemudian dibandingkan dengan penjadwalan Round Robbin umum dalam bentuk grafik.

2. Definisi Scedulling

Penjadwalan proses merupakan kumpulan kebijaksanaan dan mekanisme di sistem operasi yang berkaitan dengan urutan kerja yang dilakukan sistem komputer[5].

Penjadwalan bertugas memutuskan proses yang harus berjalan dan kapan dan selama berapa lama proses itu berjalan. Kriteria untuk mengukur dan optimasi kinerjasebuah penjadwalan adalah :*Fairnes* (adil), *Eeficiency* (efisiensi), *Response time* (waktu tanggap); yang terbagi dalam 2 sistem, yaitu Sistem interaktif dan Sistem waktu nyata, *Turn around time*; dengan persamaan waktu eksekusi + waktu menunggu serta *Throughput*[3].Tujuan penjadwalan yaitu :

1. Menjamin tiap proses mendapat pelayanan dari pemroses yang adil.
2. Menjaga agar pemroses tetap dalam keadaan sibuk sehingga efisiensi mencapai maksimum. Pengertian sibuk adalah pemroses tidak menganggur, termasuk waktu yang dihabiskan untuk mengeksekusi program pemakai dan sistem operasi.
3. Meminimalkan waktu tanggap.
4. Meminimalkan turn arround time.
5. Memaksimalkan jumlah job yang diproses persatu interval waktu. Lebih besar angka *throughput*, maka akan lebih banyak kerja yang dilakukan sistem [5].

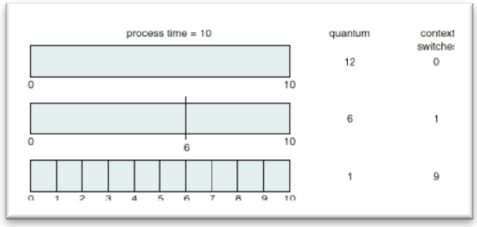
3. Fetching data pada CPU

Fetching adalah proses pembacaan byte instruksi dari antrian memory menggunakan program counter (PC) sebagai alamat memory. Instruksi tersebut kemudian diekstrak menjadi 2 porsi 4-bit yang dikenal sebagai icode (*instruction code*) dan ifun (*instruction function*) [1]. Secara keseluruhan ada 5 langkah proses dalam organisasi proccessor, yaitu *fetching* (pembacaan data), d*ecode* (pengalamatan), *execute* (pengolagan/eksekusi), *memory* (penyimpanan) dan *write back* (kembali ke register file)[1]. Antrian yang tersusun dalam processor ketika sebuah komputer On adalah antrian real time yang berasal dari semua operasi pengolahan data yang dikerjakan. Kelima proses ini berjalan secara paralel dalam mengeksekusi setiap pengolahan data dalam CPU. Apabila proses *fetching* lambat, maka proses selanjutnya juga ikut lambat dan dapat mempengaruhi *performance* sebuah CPU (*Central Processing Unit*). Padahal dalam menggunakan komputer, setiap user akan menginginkan *performance* komputer yang cepat dalam pekerjaannya.

4. Penjadwalan Round-Robin

Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar time quantum. Jika time quantum-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 quantum.

Algoritma ini sepenuhnya bergantung besarnya time quantum. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma first come first served. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang.



Gambar 4.1. Penjadwalan Round-Robin

Permasalahan pada Round Robin adalah menentukan besarnya *time quantum*. Jika *time quantum* yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesaidalam 1 quantum. Hal ini tidak baik karena akan terjadi banyak switch, padahal CPU memerlukan waktu untuk beralih dari suatu proses ke proses lain (*context switches time*). Sebaliknya, jika *time quantum* terlalu besar, algoritma Round Robin akan berjalan seperti algoritma *first come first served*. Time quantum yang ideal adalah jika 80% dari total proses memiliki CPU *burst time* yang lebih kecil dari 1 *time quantum*[4].

Sebelum dibuat pengembangan penjadwalan round robbin dalam algoritma lain, penulis terlebih dahulu akan memaparkan algoritma umum yang biasa diberlakukan dalam penanganan proses CPU.

Algoritma penjadwalan round robbin diberikan sebagai berikut [3]:

a. Mengatur semua proses di dalam antrian
b. Memberikan time quantum (q) untuk membatasi waktu proses
c. Jika waktu proses selesai, proses ditunda dan ditambahkan pada antrian ready.
d. Jika suatu proses memiliki burts time > time quantum, maka proses akan melepaskan CPU setelah selesai dan CPU akan segera dapat digunakan oleh proses selanjutnya.
e. Jika suatu proses memiliki burst time < time quantum, maka proses tersebut akan dihentikan jika sudah mencapai waktu quantum dan selanjutnya mengantri kembali pada posisi ekor dari daftar antrian. CPU kemudian menjalankan proses berikutnya.
f. Tidak ada proses yang menunggu lebih dari (n-1)q unit waktu.
g. Jika q besar maka round robbin menggunakan aturan FCFS, sedangkan jika q kecil, maka menggunakan mekanisme konteks *switch*.

Untuk menerapkan Round Robbin pada proses *fetching*, disediakan sebuah program dengan 5 proses sebagai berikut :

```
P1 : push1 %edi
P2 : push1 %esi
P3 : imul 16(%ebp), %eax
P4 : leal 0(,%eax,4), %ecx
P5 : mov1 %ebx, %edx
```

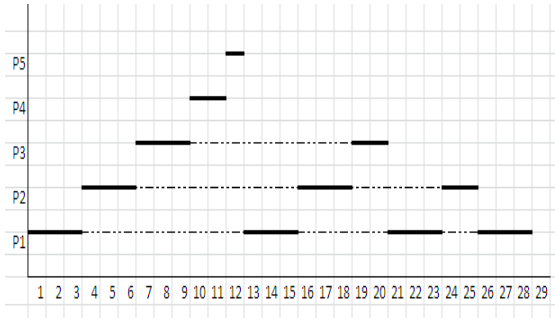
Jika diolah dengan algoritma Round Robbin umumsebagai berikut :

a. Eksekusi dengan Burst Time (BT)saja.

Artinya ketika kelima program masuk dalam antrian sekaligus/tanpa waktu kedatangan (AT) dan quantum (q) = 3.

Proses	Burst Time (BT)
P1	12
P2	8
P3	5
P4	2
P5	1

Grafik *Huntt* sebagai berikut :



Dari grafik di atas, diperoleh rata-rata waktu tunggu dan rata-rata *turn around time* sebagai berikut :

P1 = (12-3)+(20-15)+(25-23) = 16
P2 = (15-6)+(23-18) = 14
P3 = (18-9) = 9
P4 = 0 = 0
P5 = 0 = 0
Jumlah = 39

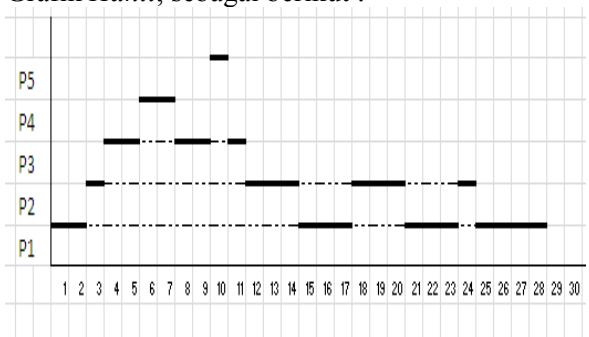
AverageWaiting Time (AWT) = 7.8
Average Turn Around Time (ATAT) = 13.4
Performance CPU = 0.128

b. Eksekusi dengan Burst Time (BT) dan waktu kedatangan (AT) yang diketahui.

Artinya kelima proses tersebut meng-antri dengan waktu kedatangan diketahui dan q = 3.

Proses	Arival Time (AT)	Burst Time (BT)
P1	0	12
P2	2	8
P3	3	5
P4	5	2
P5	9	1

Grafik *Hantt*, sebagai berikut :



Dari grafik di atas, diperoleh rata-rata waktu tunggu dan rata-rata turn around time sebagai berikut :

P1 = (14-2)+(20-17)+(24-23) = 16
P2 = (11-3)+(17-14) + (23-20) = 24

P3 = (7-5) + (10-9) = 3
P4 = 0 = 0
P5 = 0 = 0
Jumlah = 43

Average Waiting Time (AWT) = 8.6
Average Turn Around Time (ATAT) = 12.2
Performance CPU = 0.116

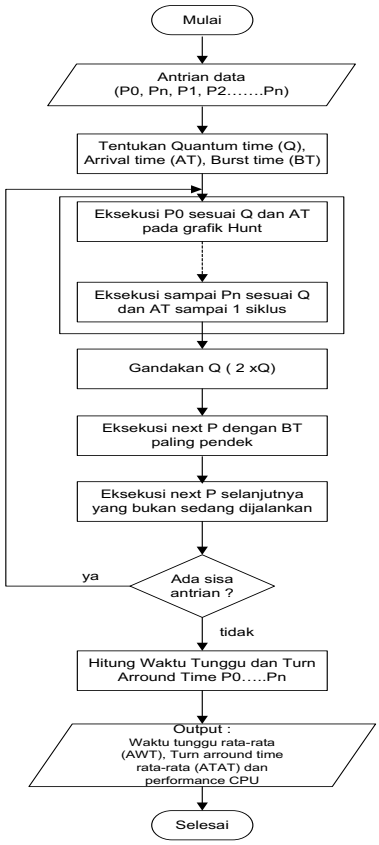
Dari kedua percobaan di atas terlihat bahwa pada percobaan (a), dengan 5 proses tanpa AT diperoleh *rata-rata waiting time* (AWT) sebesar 7.8 satuan waktu, rata-rata *turn around time* (ATAT) sebesar 13.4 satuan waktu dan performance CPU sebesar 0.128. Sedangkan pada percobaan (b) dengan memperhitungkan faktor AT diperoleh rata-rata *waiting time*(AWT) sebesar 8.6 satuan waktu, ATAT sebesar 12.2 satuan waktu dan p*erformance* CPU-nya sebesar 0.116. Jika proses yang berada dalam antrian semakin banyak, maka akan terjadi peningkatan AWT dan ATAT sehingga *performance* CPU akan menjadi lebih lambat.

5. Pengembangan Penjadwalan Round-Robin.

Untuk mengurangi/menurunkan rata-rata *waiting time* (AWT) dan turn around time (ATAT)pada 5 proses program di atas, maka dibangun pengembangan penjadwalan Round Robbin sebagai berikut:

Proses dimulai dengan meninjau antrian data P_1, P_2,…P_n. Kemudian menentukan quantum time (q) dan menerapkan pada setiap proses melalui Grafik Huntt (*huntt chart*).

Eksekusi proses P_1, P_2, ...$P_n$dalam antrian berdasarkan q yang diberikanhingga selesai satu siklus.Jika telah selesai, gandakan quantum timemenjadi 2.q dan eksekusi sisa proses (P_1, P_2, ...P_n) yang memiliki *burst time* (BT) paling pendek dari antrian.
Selanjutnya, pilih proses terpendek berikutnya dan eksekusi, dengan catatan bahwa proses yang dipilih ini bukan termasuk proses yang telah dijalankan sebelumnya dalam siklus yang sedang berjalan.Jika masih ada sisa antrian, maka ulangi dari awal.



Gambar 5.1. Pengembangan Round Robbin

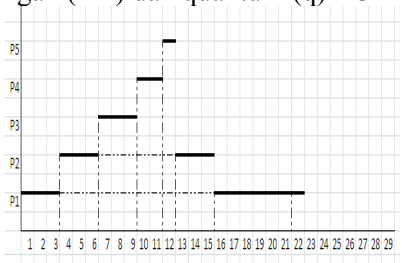
Jika algoritma iniditerapkan pada 5 proses sebagaimana pada bagian (5), maka aktifitasnya sebagai berikut :

```
P1 : push1 %edi
P2 : push1 %esi
P3 : imul 16(%ebp), %eax
P4 : leal 0(,%eax,4), %ecx
P5 : mov1 %ebx, %edx
```

Jika diolah dengan algoritma pengembangan Round Robbin sebagai berikut :

a. Proses *real time* dengan *Burst Time* (BT) saja.

Artinya ketika kelima program masuk dalam antrian sekaligus/tanpa waktu kedatangan (AT) dan quantum (q) = 3



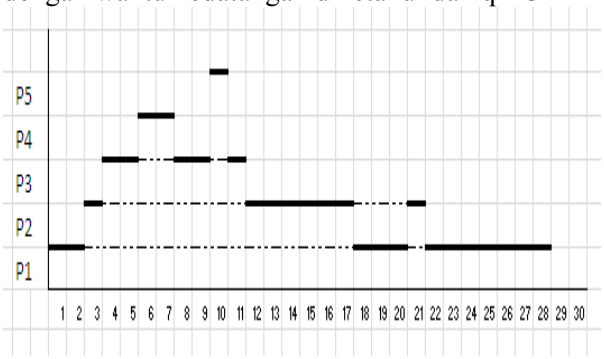
Dari grafik di atas, diperoleh rata-rata *waiting time* dan rata-rata *turn around time* sebagai berikut :

P1 = (15-3) = 12
P2 = (12-6) = 6
P3 = 0 = 0
P4 = 0 = 0
P5 = 0 = 0
Jumlah = 18

Average Waiting Time (AWT) = 3.6
Average Turn Around Time (ATAT) = 8
Performance CPU = 0.278

b. Eksekusi dengan Burst Time (BT) dan waktu kedatangan (AT) yang diketahui.

Artinya kelima proses tersebut meng-antri dengan waktu kedatangan diketahui dan q = 3



Dari grafik di atas, diperoleh rata-rata *waiting time* dan rata-rata *turn around time* sebagai berikut :

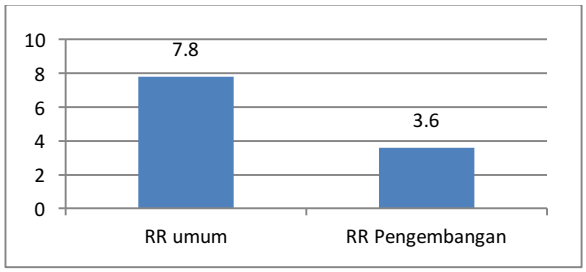
P1 = (17-2)+(21-20) = 16
P2 = (11-3)+(20-17) = 11
P3 = (7-5)+(10-9) = 3
P4 = 0 = 0
P5 = 0 = 0
Jumlah = 30

Average Waiting Time (AWT) = 6
Average Turn Around Time (ATAT) = 11.6
Performance CPU = 0.167

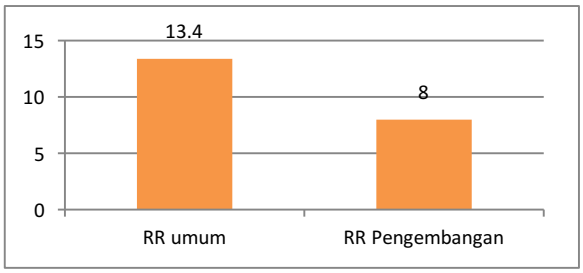
Dari kedua percobaan di atas terlihat bahwa pada percobaan (a), dengan 5 proses tanpa AT diperoleh rata-rata *waiting time* (AWT) sebesar 3.6 satuan waktu, rata-rata *turn around time* (ATAT) sebesar 8 satuan waktu dan *performance* sebesar 0.278. Sedangkan pada percobaan (b) dengan memperhitungakan faktor AT diperoleh rata-rata *waiting time*(AWT) sebesar 6 satuan waktu, ATAT sebesar 11.6 satuan waktu dan performance CPU sebesar 0.167.

6. Analisa

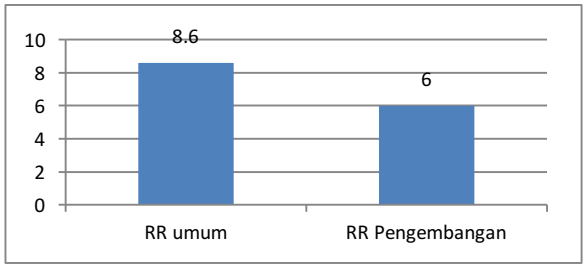
Jika Round Robbin umum dan Pengembangan ini dibandingkan dalam grafik, maka diperoleh sebagai berikut :

1. Grafik perbandingan *waiting time*(AWT) dan *turn around time* tanpa *arival time* (AT).
2. 

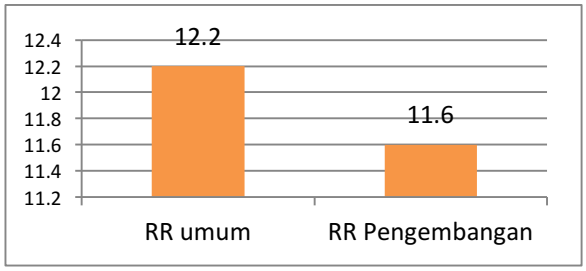
3. Grafik perbandingan *turn around time* (ATAT) tanpa *arival time* (AT)



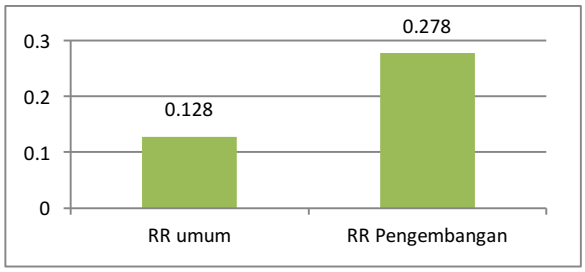
4. Grafik perbandingan *waiting time* (AWT) dan *turn around time* menggunakan *arival time* (AT).



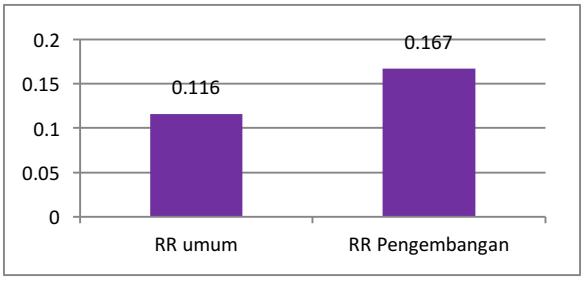
5. Grafik perbandingan *turn around time* (ATAT) menggunakan *arival time* (AT)



6. Grafik perbandingan performance CPU tanpa*arival time* (AT)



7. Grafik perbandingan performance CPU menggunakan*arival time* (AT)



Grafik (1) dan (2) adalah grafik AWT dan ATAT ketika *arival time* tidak diperhitungkan baik pada RR umum dan RR pengembangan. Terlihat bahwa ketika menerapkan Round Robbin pengembangan AWT turun dari 7.8 menjadi 3.6 satuan waktu sedangkan ATAT turun dari 13.4 ke 8satuan waktu. Grafik (3) dan (4) adalah grafik AWT dan ATAT ketika arival time diperhitungkan. Terlihat bahwa ketika menerapkan Round Robbin pengembangan AWT turun dari 8.6 menjadi 6 satuan waktu sedangkan ATAT turun dari 12.2 ke 11.6 satuan waktu. Grafik (5) dan (6) adalah grafik perbandingan *performance* CPU tanpa *arival time* dan menggunakan *arival time*. Terlihat bahwa ketika tidak menerapkan *arival time*, *performance* CPU meningkat dari 0.128 menjadi 0.278. Sedangkan ketika menerapkan *arival timeperformance* CPU meningkat dari 0.116 menjadi 0.167.

Perbandingan dengan pengujian sebelumnya yang dilakukan oleh Ajit Singh, Priyanka Goyal dan Sahil Bahtra (2012) adalah sebagai berikut :

Pengujian sebelumnya	Pengujian sekarang
Menjelaskan tahapan RR pengembangan dalam 3 phase saja.	Menjelaskan algoritma lebih rinci hingga output yang mendukung *performance* CPU.
Proses diuji menggunakan *Gantt chart*.	Proses diuji menggunakan *Huntt chart*.
Parameter yang diuji adalah jumlah *switching*, AWT dan ATAT.	Parameter yang diuji adalah AWT, ATAT dan *Performance* CPU
Kesimpulan yang diambil tidak menyertakan angka pasti hasil pengujian.	Kesimpulan dengan menyertakan angka pasti pengujian (AWT, ATAT dan *Performance* CPU)

7. Kesimpulan

Kesimpulan dari laporan ini adalah :

1. Penjadwalan Round Robin pada dasarnya menggunakan *time sharing* dengan setiap proses mendapatkan waktu CPU (quantum).
2. Pengolahan proses *real time* CPU dapat dilakukan dengan menerapkan algoritma pengembangan RR. Hasil pengujian telah terjadi penurunan AWT dan ATAT yang akan mempengaruhi terjadinya peningkatan *performance* CPU.
3. Dengan menerapkan algoritma pengembangan Round Robbin, telah berhasil meningkatkan *performance* CPU ketika melakukan *fetching* data.

8. Daftar Pustaka

[1] Randal E. Bryant, David R. O'Hallaron, *Computer System A Programmer's Perspective*, Prentice Hall, 2003, 281

[2] Ajit Singh, Priyanka Goyal, Sahil Bahtra, *An Optimized Round Robin Sceduling algorithm for CPU sceduling*, International Journal on Computer Science and Engineering Vol. 02 No 07 Tahun, 2012

[3] Abbas Noon, Ali Kalakech, Seifedine Kadry, *A New Round Robin Based Sceduling Algorithm for Operating System:Dinamyc Quantum using the mean average*, Faculty of Business Lebanase University.

[4] C. Yaashuwanth, R. Ramesh, *A New Scheduling algorithm for real time task, International* Journal of computer science and invormation security, vol 6 no 2, 2009.

[5] Abas Ali Pangera, Dony Ariyus, Sistem Operasi, ANDI Yogyakarta, 2010.